Approximative Indexierungstechnik für historische deutsche Textvarianten

Heller, Markus

Historical Social Research, Vol. 31 — 2006 — No. 3, 288-307

Approximative Indexierungstechnik für historische deutsche Textvarianten

Markus Heller [*]

Abstract: Historical documents have specific properties which make life hard for traditional information retrieval techniques. The missing notion of orthography and a general high degree of variation in the phonetic-graphemic representation, as well as in derivational morphology obstruct the possibility to find documents upon the entry of a modern word as the search term. The following paper gives an overview of existing string approximation technologies as used in bioinformatics, but also of phonetic approximation algorithms. It proposes an architecture of combining both notions, while using Jörg Michael' phonet program to deduct from graphemes to a phonetic representation and a levenshtein automaton to allow for fast approximative matching. The final part of the paper evaluates the suitability of the approach, while using the levenshtein algorithm in its non-automaton-based implementation.

1. Einleitung

In der deutschen Sprache lässt sich seit den ersten schriftlichen Überlieferungen eine sehr reichhaltige Dialektlandschaft nachweisen. Die Erfindung des Buchdrucks und die damit mögliche Verbreitung der Lutherbibel führte im 16. Jahrhundert zu ersten nennenswerten Ansätzen, eine normative Hochsprache zu definieren. Doch erst durch die Wörterbücher von Johann Christoph Adelung 1774-86 und der Brüder Grimm wurde die Grundlage für eine orthographische Normierung der deutschen Sprache gelegt. So war die deutsche Sprache seit Beginn der von uns betrachteten Epoche des Neuhochdeutschen, einschließlich des Frühneuhochdeutschen, regionalen Unterschieden, aber auch diachronen Veränderungen unterworfen. Diese Unterschiede und Veränderungen finden

[*] Address all communications to: Markus Heller, Schellingstr. 43, 80799 München, Germany; e-mail: heller@cis.uni-muenchen.de.

auf allen Ebenen der Sprache statt, also auf den Ebenen der Phonetik, der Morphologie, der Lexik, der Syntax, der Semantik, bis hin zu satzübergreifenden Phänomenen.

Unser Ziel ist, einen editierten, diachronen Korpus an neuhochdeutschen Texten durchsuchbar zu machen, wobei wir davon ausgehen, dass die Texte mit XML ausgezeichnet wurden, aber von unterschiedlichen Editoren stammen und daher auch hinsichtlich der verwendeten XML-Dialekte heterogen sein können.

Da ein Benutzer mit normalen Deutschkenntnissen die Texte der neuhochdeutschen Epoche beinahe umfassend lesen und verstehen kann, gehen wir davon aus, dass die Eingabe bei einer entsprechenden Suchmaschine in modernem Deutsch erfolgt. Wir erwarten, dass auch nicht ganze Phrasen, sondern ähnlich wie bei Google einzelne Begriffe oder Begriffskombinationen eingegeben werden. Eine Suchmaschine für neuhochdeutsche Texte muss also in der Lage sein, von zeitgenössischen Wörtern in aktuellem Deutsch auf deren Formen im Korpus zu schließen. In diesem Sinne handelt es sich bei der Problemstellung noch nicht um Cross-Language Information Retrieval[1], jedoch auch nicht mehr um monolinguales, boolsches Retrieval, bei dem von einer maschinell eindeutig zu beantwortenden Ja/Nein-Antwort ausgegangen werden kann.

Die gängige Suchtechnik beruht jedoch im Wesentlichen auf der Annahme der Identität des Eingabeworts mit dem zu findenden Wort. Das Ziel unseres Projekts „DING“[2] besteht nun darin, einen approximativen Indexer für XML-Korpora zu entwickeln, der vor allem in der Lage ist, orthographisch nicht normierte und unterschiedlich tief ausgezeichnete Textbestände erschließbar zu machen. Der Indexer ermöglicht somit einen modernsprachlichen Zugriff auf Informationen, die nicht modernsprachlich abgelegt sind, d.h. er ist ein Instrument für die Arbeit von Historikern mit originalen Quellen.[3]

Die Qualitätskriterien zur Bewertung von Information Retrieval Technologien lauten Precision und Recall:

- *Precision* beschreibt das Verhältnis der Anzahl der relevanten Dokumente zu der Anzahl der insgesamt gefundenen Dokumente.
- *Recall* beschreibt die Anzahl der relevanten gefundenen Dokumente zu der Anzahl aller auf die gestellte Anfrage relevanten Dokumente.

---

[1] Oard hat eine sehr gute Übersicht zu Algorithmen und Themen zum Thema „Cross Language Information Retrieval“ zusammengestellt: <http://www.glue.umd.edu/~oard/research.html> (Letzte Einsichtnahme: 23.05.2006).

[2] DING steht für „Das ist nicht Google“. Das Projekt an der Ludwig-Maximilians-Universität entsteht im Kontext der Arbeit an einem umfassenden Informationssystem zu Urkunden des Mittelalters und der Frühen Neuzeit (vgl. u.a. die Charters Encoding Initiative <http://www.cei.lmu.de/>, letzte Einsichtnahme 2.6.2006).

[3] Vgl. dazu: Georg Vogeler: Fachspezifische Indexierung von historischen Dokumenten. Quellen zwischen Zeichenketten und Information – Beispiel Urkunden, in: Forschung in der digitalen Welt. Sicherung, Erschließung und Aufbereitung von Wissensbeständen, hg. v. R. Hering, Hamburg 2006, im Druck.

Wenn also Dokumente gefunden werden sollen, die sehr wohl noch als relevant betrachtet werden, dann müssen Verfahren gefunden werden, die den Recall erweitern, die also mehr Dokumente aus der Menge aller im Korpus befindlichen relevanten Dokumente finden. Da solche „Aufweichungen“ meist auch zu einer schlechteren Trennschärfe führen, sinkt häufig die Precision, also das Verhältnis der Anzahl der relevanten zu der Anzahl der irrelevanten Dokumente. Es wird häufig mehr irrelevantes Material gefunden, wenn das Auswahlkriterium weicher gestaltet wird.

Im vorliegenden Beitrag soll im Wesentlichen konzeptionell untersucht werden, welche Möglichkeiten bestehen, auf die besonderen Eigenschaften eines neuhochdeutsch-diachronen und transdialektalen Korpus anhand bestehender approximativer Indexierungsverfahren einzugehen. Besondere Aufmerksamkeit sollen dabei phonetische Approximationsalgorithmen erfahren, da die reinen Stringverfahren zwar an und für sich sehr gute Ergebnisse liefern, jedoch in Bezug auf einen orthographisch nicht normierten Korpus schnell überfordert sind.

2. Methoden der Approximation

Grundsätzlich haben alle Ansätze zur Erweiterung des Recall das Ziel, mehr Kandidaten als Treffer zu klassifizieren als jene, die durch die Identität des Eingabeworts mit dem gesuchten Begriff zustande kommen. Dies kann im Wesentlichen durch drei Methoden erreicht werden:

1) Es kann ohne Blick auf die den zu durchsuchenden Daten innewohnende Regelhaftigkeit eine gewisse, begrenzte Anzahl an Fehlern zugelassen werden.
2) Es besteht die Möglichkeit, ohne Blick auf die zu durchsuchenden Daten mögliche alternative Varianten des Suchbegriffs zu generieren und diese als Suchbegriffe, etwa in einer Datenbank zu nutzen. Hierfür ist es nötig, die Regelhaftigkeit der Abweichungen im Korpus zu untersuchen, um nicht völlig unzutreffende Varianten zu generieren.
3) Die weitere Methode besteht darin, die Regelhaftigkeit der Daten zu berücksichtigen und darauf basierend Äquivalenzklassen zu definieren, mit denen bestimmte begründete Abweichungen zugelassen werden.

Bei der ersteren Methode besteht die Schwierigkeit in der Definition der Fehlergrenze. Man akzeptiert alle gefundenen Zeichenketten, die weniger als eine bestimmte Anzahl von Unterscheidungen zu der Zeichenkette aufweisen, mit der man die Suche gestartet hat. Will man nun den Recall verbessern, indem man die Fehlergrenze erhöht, so akzeptiert man notgedrungen auch jene Trefferkandidaten, die bei genauerem Hinsehen falsch sind, was zur Folge hat, dass die Precision sich verschlechtert.

Die zweite Methode wurde von Andrea Ernst-Gerlach und Norbert Fuhr spezifiziert und stellt den aktuellen Forschungsstand dar.[4] Ernst-Gerlach und Fuhr leiten aus einem vorliegenden Korpus im Vergleich zur normierten, modernen Orthographie die Veränderungen zwischen alten und neuen Schreibungen ab und modellieren diese. Aus den maschinell gewonnenen Veränderungsregeln generieren sie mögliche historische Schreibweisen. Die gewonnenen Erkenntnisse lassen sich bislang jedoch nur auf den Nietzsche-Korpus mit Texten aus den Jahren zwischen 1865 und 1945 anwenden. Regionale Schreibungen und dialektbedingte Variationen sind in diesem Ansatz nicht berücksichtigt. Die Frage, ob durch ein rein induktives Verfahren auch zuvor nicht identifizierte Schreibvarianten generiert werden können, muss methodisch hinterfragt werden.

Ähnlich wie Ernst-Gerlach und Fuhr sehen wir auch die Notwendigkeit, auf die Eigenschaften der Daten einzugehen, wählen aber den dritten Weg, indem wir zwischen der Varianz der Phonetik und der Varianz der Schreibung unterscheiden und beide Probleme gezielt einzeln adressieren. Zu diesem Zweck liegt es nahe, sowohl rein mathematisch approximative Stringverfahren, aber auch phonetische Verfahren zu untersuchen. Unser Ziel ist, ein Verfahren zu skizzieren, das beide Ansätze kombiniert.

2.1. Stringverfahren zur Approximation

In der Literatur sind zahlreiche Ansätze für die Definition der Ähnlichkeit zweier Zeichenketten bekannt.[5] Zumeist wurden sie nicht mit Blick auf die Untersuchung der Ähnlichkeit zweier Wörter entwickelt, sondern im Rahmen der Bioinformatik-Forschung, in der häufig die Fragestellung zu beantworten ist, ob zwei Basenpaare eines Genoms Verwandtschafts- oder Ähnlichkeitsmerkmale aufweisen. Der ursprüngliche Kontext der Verfahren besteht zumeist lediglich aus den beiden Zeichenketten, die als solche unmittelbar vorliegen. Umstände, bei denen eine der beiden Zeichenketten nicht vorliegt, und bei denen erst in einer Menge an Schlüsseln danach gesucht werden muss, wurden meist nicht bedacht. Ein Beispiel hierfür wäre eine Genomsequenz, nach der in einem großen Pool an Genomdaten gesucht wird. Dabei kann der Pool linear abgearbeitet werden, wobei die gesuchte Zeichenkette und der Kandidat vorliegen, ehe ein qualifizierender Vergleich bezüglich der Ähnlichkeit errechnet wird.[6] Durch den linearen Ansatz sind diese Prüfverfahren trotz ihrer hohen

[4] Andrea Ernst-Gerlach & Norbert Fuhr. Generating Search Term Variants for Text Collections with Historic Spellings. 28th European Conference on Information Retrieval Research (ECIR 2006). <http://www.is.informatik.uni-duisburg.de/bib/pdf/ir/Ernst_Fuhr:06.pdf> (Letzte Einsichtnahme: 30.05.2006).

[5] Einen umfassender Überblick über etablierte Ansätze: Gonzalo Navarro. A Guided Tour to Approximate String Matching. ACM Comput. Surv. 33, 1 (03/2001). S. 31-88.

[6] Eine exzellente Einführung in zahlreiche lineare approximative- und nicht-approximative Matchingverfahren: Gonzalo Navarro & Mathieu Raffinot. Flexible Pattern Matching in

Performanz sehr zeitintensiv: Sie arbeiten alle Daten sequentiell ab und enden nach der Überprüfung des letzten Strings. Es ist jedoch möglich, statt des linearen Verarbeitungsansatzes einen strukturellen zu wählen. Er gleicht dem Nachschlagen in einem Telefonbuch: Voraussetzung hierfür ist jedoch, dass die zu durchsuchenden Daten in einer entsprechenden Datenstruktur vorverarbeitet wurden. Doch dazu später mehr.

Der bekannteste lineare Approximationsalgorithmus stammt von Fred Damerau und Vladimir Levenshtein.[7] Er definiert die Messbarkeit von Ähnlichkeit dergestalt, dass alle erforderlichen Einfügungen, Löschungen und Ersetzungen gezählt werden, bis beide Zeichenketten identisch sind. Das Verfahren beruht meist auf Methoden aus der dynamischen Programmierung, d.h., dass zur Anfragezeit eine Matrix mit den entsprechenden Distanzmaßen aufgebaut und anschließend abgearbeitet wird.

Das Verfahren von Hamming unterscheidet sich vom Levenshtein-Verfahren insoweit, als nur Vertauschungen, aber nicht Einfügungen oder Löschungen erkannt werden. Es ist deshalb nur für Zeichenketten gleicher Länge geeignet, da Verschiebungen nach einer Löschung oder einer Einfügung stets zu einer Mehrdistanz in Höhe der Anzahl der Zeichen führen, die je um eine Position verschoben werden müssen. Ursprünglich war es dazu gedacht, die Anzahl von Fehlern in einer Datenübertragung fester Länge zu messen.[8]

Das N-Gramm-Verfahren[9] zerlegt die beiden zu untersuchenden Zeichenketten in N-Gramme, meist um Bi- oder Trigramme. Die Annahme besteht darin, dass die beiden Zeichenketten umso ähnlicher zueinander sind, je mehr N-Gramme übereinstimmen. Das N-Gramm-Verfahren birgt im Vergleich zu dem Levenshtein-Verfahren den Vorteil, dass die N-Gramme direkt indexierbar sind.[10]

Strings. Practical On-Line Search Algorithms for Texts and Biological Sequences. Cambridge: CUP, 2002.

[7] Vladimir I. Levenshtein: Binary codes capable of correcting deletions, insertions, and reversals, in: Doklady Akademii Nauk SSSR 163 (1965), S. 845-848 (Russisch). Englische Übersetzung in: Soviet Physics Doklady, 10 (1966), S. 707-710. Obwohl Fred Damerau Levenshtein mit der Publikation eines identischen Verfahrens zwei Jahre zuvorkam, ist es unter dem Namen Levenshteins bekannt geworden. Häufig ist aber auch die Bezeichnung „Damerau-Levenshtein-Algorithmus“ zu finden. Vgl. Fred J. Damerau. A Technique for Computer Detection and Correction of Spelling Errors. In. Communications of the ACM. Vol 7, Nr. 3 (03/1964). S. 171-176. <http://dalcin.org/referencias/pdf/325.pdf> (Letzte Einsichtnahme 30. Mai 2006).

[8] Richard Hamming: R.W. Hamming. Error detecting and error correcting codes. Bell System Tech. J. 29, 1950. S. 147-160.

[9] Das Verfahren ist auch unter dem Namen „Q-Gramm-Verfahren“ bekannt und wurde 1992 spezifiziert: Esko Ukkonen. Approximate string matching with q-grams and maximal matches. Theoretical Computer Science, Vol. 92, Iss. 1, 1992. S. 191-211.

[10] S. Burkhardt & A. Crauser & P. Ferragina & H.-P. Lenhof & E. Rivals, M. Vingron. q-gram based database searching using a suffix array. In: RECOMB99, ACM, NewYork, 1999. S. 77-83.

Ein weiteres Verfahren ist die sog. Longest-Common-Subsequence-Distanz, bei dem Einfügungen und Löschungen erlaubt sind. Es misst die Länge des längsten gemeinsamen Teilstrings, unter der Voraussetzung, dass beide Teilzeichenketten die Reihenfolge der Zeichen beibehalten. Die Distanz zwischen den beiden Strings errechnet sich aus der Anzahl der unterschiedlichen Zeichen.[11]

Die sog. Episode-Distanz, die im Approximations-Kontext ebenfalls erwähnt wird, ist für unser Vorhaben ungeeignet, da sie ausschließlich Einfügungen erlaubt. Der Unterschied zwischen einer modernen und einer historischen Schreibung eines Worts kann aber nicht ausschließlich mit Einfügungen beschrieben werden.[12]

Wie eingangs erwähnt, finden die rein mathematischen Distanzverfahren besonders in der Bioinformatik Anwendung. Distanzmaße sind umso aussagekräftiger, je länger die zu untersuchenden Zeichenketten sind. Im Vergleich zu den phonetischen Approximationsverfahren haben sie bei modernen Namensbeständen deutliche Stärken.[13] Zobel und Dart kommen sogar zu dem Ergebnis, dass phonetische Verfahren den reinen Stringverfahren in Bezug auf kontemporäre Lexika deutlich unterlegen sind.[14]

Allerdings wird anhand des Pronomens „uns“ im Vergleich zu der sehr häufigen frühneuhochdeutschen Schreibung „vnnß“ deutlich, dass reine mathematische Verfahren an ihre Grenzen stoßen. Es ist allgemein bekannt, dass Levenshtein-Verfahren bei einem Korpus an menschlicher Sprache bis zu einer Editierdistanz von drei Operationen brauchbare Ergebnisse liefern.[15] Darüber hinaus steigt die Anzahl der fehlerhaften Treffer übersignifikant an. Es scheint also wünschenswert, das Levenshtein-Verfahren mit linguistisch rechtfertigbaren Äquivalenzklassen anzureichern, damit linguistisch begründete Varianzen nicht zu unnötigen Fehlerzählern führen.

2.2. Phonetische Approximationsverfahren

Die Entwicklung phonetischer Approximationsverfahren begann mit der Notwendigkeit, ähnliche Schreibungen von Personennamen identifizieren zu kön-

[11] S. Needleman & C. Wunsch. A general method applicable to the search for similarities in the amino acid sequences of two proteins. Journal of Molecular Biology 48, 1970. S. 444-453. Vgl. auch A. Apostolico & C. Guerra. The Longest Common Subsequence problem revisited. Algorithmica 2, 1970. S. 315-336.

[12] G. Das & R. Fleisher & L. Gasieniek & D. Gunopulos & J. Karkainen. Episode matching. In Proceedings of the 8th Annual Symposium on Combinatorial Pattern Matching (CPM ‘97). LNCS, vol. 1264, Springer-Verlag, Berlin, 1997. S. 12-27.

[13] Vgl. Rainer Schnell & Tobias Bachteler & Stefan Bender. A Toolbox for Record Linkage. Austrian Journal of Statistics 33 (2004), Nr. 1 & 2. S. 125-133.

[14] Justin Zobel & Philip Dart. Finding Approximate Matches in Large Lexicons.Software-Practice and Experience, Vol. 25(3), 1995. S. 331-345.

[15] Hierbei handelt es sich um einen Erfahrungswert.

nen und war damit seit jeher eng mit der Phonotaktik der verwendeten Sprache verbunden. Auch wenn unser Projekt explizit nicht auf Personennamen beschränkt ist, so sind die Verfahren rund um das Problem der „Record Linkage" für uns sehr interessant, da sich in der Graphie insbesondere von Familiennamen ältere Sprachstände verfestigt haben, die Suche in Personennamen häufig um einen Vergleich zwischen einer neueren und einer älteren Namensform handelt. Wir nehmen an, dass die Erkenntnisse aus der Arbeit mit Personennamen sich direkt auf den Rest des Lexikons übertragen lassen.

Der wohl bekannteste, und älteste Algorithmus zur Feststellung von Namensähnlichkeit ist der Soundex-Mechanismus[16], der bereits in den Jahren 1918-1922 beschrieben und patentiert wurde. Im Wesentlichen werden alle Vokale eliminiert und die Konsonanten gewissen Gruppen zugeordnet, sodass sich für jedes Wort ein Code ergibt. Der Algorithmus ist sehr Präfix-zentriert, denn der Soundex-Code eines Wortes wird nach der vierten Stelle abgeschnitten. Die Hauptpunkte der Kritik an diesem Verfahren bestehen einerseits in der starken Englisch-Lastigkeit der Äquivalenzdefinitionen, der reinen zeichenorientierten Berechnung der Ähnlichkeit und der Tatsache, dass Wortsuffixe ab einer bestimmten Stelle nicht mehr berücksichtigt werden.[17]

Es existieren verschiedene Weiterentwicklungen zum Soundex-Verfahren, so etwa das „New York State Identification and Intelligence System" (NYSIIS), das durch die Einführung von bestimmten Translationsregeln einen Präzisionsgewinn von 2.7% gegenüber den ursprünglichen Soundex-Verfahren erzielt.[18] Randy Daitch und Gary Mokotoff haben eine Variante zum Soundex-Verfahren vorgestellt, das besonders die phonetischen Eigenheiten der Namen osteuropäisch-jüdischer Einwanderer berücksichtigt.[19]

Die genannten Defizite des Soundex-Verfahrens haben Lawrence Phillips 1990 dazu veranlasst, das Metaphone-Verfahren[20] zu entwickeln, welches die Soundex-Mechanik durch zahlreiche linguistische Erkenntnisse verbessert und

---

[16] Entwickelt von Robert Russell und Margaret Odell in den Jahren 1918 und 1922 und patentiert durch die U.S.-Patente 1261167 und 1435663.
Vgl. <http://patft.uspto.gov/netacgi/nph-Parser?patentnumber=1,261,167> und <http://patft.uspto.gov/netacgi/nph-Parser?patentnumber=1,435,663> (beide letzte Einsichtnahme 30. Mai 2006). Siehe auch: Donald E. Knuth, The Art of Computer Programming. Vol. 3, 2nd ed. Boston: Addison-Wesley, 1998. S. 394-395.

[17] Ein populäres Beispiel für die Defizite in der Präzision des Soundex-Verfahrens besteht im Vergleich der Soundex-Codes der Terme (Britney) „Spears" und „Superzicke", welche identisch sind („S162").

[18] Taft, R.L. Name Searching Techniques. Albany, N.Y.: Bureau of System Development, 1970.

[19] Gary Mokotoff & Sallyann Amdur Sack. Where Once We Walked. Avotaynu, 2002. S. 567-569. Im Internet: <http://www.avotaynu.com/soundex.html> (Letzte Einsichtnahme 27. Mai 2006).

[20] Lawrence Philips. Hanging on the Metaphone. Computer Language, Vol. 7, No. 12 (December), 1990. Siehe auch: Andrew Binstock, John Rex. Metaphone: A Modern Soundex. Practical Algorithms For Programmers. Addison-Wesley, 1995. S. 160-169.

erweitert. Im Vergleich mit dem ursprünglichen Soundex-Verfahren bestehen auch Gemeinsamkeiten, insofern als auch bei Metaphone Vokale nach dem ersten Zeichen ignoriert und verschiedene Zusammenfassungen bestimmter Zeichen vorgenommen werden (beispielsweise „d“ und „t“). Allerdings werden Zeichen nicht isoliert betrachtet, sondern stets in ihrem englischen, phonetischen Kontext. Insgesamt reduziert Metaphone die Zeichen des Eingabeworts mit zahlreichen Transformationsregeln auf 16 konsonantische Repräsentationen (B X S K J T F H L M N P R O W Y).[21]

Da auch dieses Verfahren letzten Endes nicht den Idealvorstellungen seines Autors entsprach, wurde es 2000 von ihm selbst überarbeitet und als „Double Metaphone“ bekannt.[22] Es zeichnet sich dadurch aus, dass es für einen Eingabestring zwei Codes ausgibt, wobei der Sekundärcode dafür gedacht ist, eine alternative Lautung darzustellen, wie sie mit Blick auf die komplexe Eigennamenlandschaft eines ehemaligen Einwanderungslandes wie der U.S.A. vonnöten ist.

Das Verfahren von Gloria Guth[23], das auch bei KLEIO[24] zum Einsatz kommt, wurde in einer Untersuchung von Lait und Randell im Vergleich zu Soundex, Metaphone und „K-Approximate String Matching“ als deutlich unterlegen befunden.[25]

Obgleich der Großteil der in der Literatur beschriebenen Verfahren sich an der englischen Phonologie orientiert, wurden auch Ansätze publiziert, die der deutschen Aussprache gerecht werden. Vor allem J. Postel[26] hat 1969 eine am Soundex-Verfahren orientierte Translationstabelle für eine Reduktion auf acht Symbole definiert, die als „Kölner Phonetik“ bekannt wurde. Hans-Peter von Reth und Hans-Jörg Schek haben sich 1977 ebenfalls mit der Frage der Ähn-

---

[21] Codebeispiele zum Metaphone-Verfahren stehen hier zur Verfügung: <http://aspell.net/metaphone/> (Letzte Einsichtnahme 26. Mai 2006).

[22] Lawrence Philips. The Double Metaphone Search Algorithm. C/C++ Users Journal, June 2000. Hier die Wiedergabe in Dr. Dobb’s Journal: <http://www.ddj.com/dept/cpp/184401251> (Letzte Einsichtnahme 26. Mai 2006). Eine im Vergleich zur ursprünglichen Implementierung von Lawrence Phillips optimierte Variante des Double-Metaphone-Verfahrens steht hier zur Verfügung: Adam Nelson. Implement Phonetic („Sounds-like“) Name Searches with Double Metaphone. <http://www.codeproject.com/string/dmetaphone1.asp> (Letzte Einsichtnahme 26. Mai 2006).

[23] Gloria J.A. Guth. Surname Spellings and Computerized Record Linkage. Historical Methods Newsletter, 10 (1976). S. 10-19. Siehe auch: D. De Bron & M. Olsen. The Guth Algorithm and the Nominal Record Linkage of Multi-Ethnic Populations. Historical Methods, 19 (1986). S. 20-24.

[24] KLEIO ist ein nichtrelationales Datenbanksystem für historische Informationsentitäten, das unter der Führung von Manfred Thaller entwickelt wurde: <http://wwwuser.gwdg.de/~mthalle2/> (Letzte Einsichtnahme: 30.05.2006).

[25] A. J. Lait & Brian Randell. An Assessment of Name Matching Algorithms. <http://www.hpdd.de/download/wb/20010416.NameMatching.pdf> (Letzte Einsichtnahme: 29. Mai 2006).

[26] H-J. Postel, Die Kölner Phonetik – Ein Verfahren zur Identifizierung von Personennamen auf der Grundlage der Gestaltanalyse in IBM-Nachrichten 19, 1969. S. 925-931.

lichkeit in der deutschen Phonologie befasst und eine Soundex-Variante entwickelt.[27]

Die jüngsten Ansätze für eine phonetische Approximation des Deutschen stammen von Jörg Michael[28], der 1988 ein als dBase-Funktion implementiertes Verfahren namens „Phonet“ vorgestellt hat. Die letzte bekannte Version wurde 2003 unter der „Lesser GNU Public License“ in der Version 1.3.4 in Form einer C-Datei publiziert und enthält über 950 kontextsensitive Transformationsregeln.[29]

In einer umfassenden, vergleichenden Untersuchung von Approximationsverfahren in Bezug auf deutsche Familiennamen hat Martin Wilz[30] das Verfahren von Jörg Michael neben einer Implementierung der Kölner Phonetik als das leistungsfähigste bezüglich Recall und Precision beschrieben.

3. Verfahren zur Feststellung von Ähnlichkeit unter Verwendung eines Index

3.1 Grundlagen der Indexierung

Ein Index besteht in erster Linie aus einer großen Menge an Schlüsseln, ähnlich wie die Namensliste in einem Telefonbuch. Diese Liste kann mitunter sehr groß sein und umfasst im einfachsten Fall alle im Korpus vorkommenden Wörter in all ihren Formen. Nun besteht die Anforderung, diese lineare Menge an Schlüsseln durchsuchbar zu machen. Ähnlich wie bei einem Telefonbuch, wo der Suchende ebenso wenig mit dem ersten Eintrag beginnt und aufhört, sobald er den gesuchten Namen gefunden hat, wird bei Softwareindexen verfahren: Ein Telefonbuch schlägt man etwa in der Mitte auf, und unterscheidet dann mittels Vergleich an einer Ordnung, dem Alphabet, ob man das Verfahren in der oberen oder in der unteren Hälfte fortsetzen soll. Dieses Verfahren aus dem täglichen Leben ist in der Informatik modellierbar und wird „Entscheidungsbaum“ genannt.

---

[27] Hans-Peter von Reth & Hans-Jörg Schek. Eine Zugriffsmethode für die phonetische Ähnlichkeitssuche. Technical Report Nr. 77.03.002. Heidelberg: IBM Scientific Center, 1977.

[28] Jörg Michael. Nicht wörtlich genommen – Schreibweisentolerante Suchroutinen in dBase implementiert. In: c’t Magazin für Computer und Technik 10 (1988). S. 126-131.

[29] Vgl. Jörg Michael. Doppelgänger gesucht. Ein Programm für die kontextsensitive phonetische Stringumwandlung. In: c’t Magazin für Computer und Technik 25 (1999). S. 252 ff. Unter folgender URL werden die Dateien zum Artikel zum Download angeboten: <http://www.heise.de/ct/ftp/99/25/252/> (Letzte Einsichtnahme: 29. Mai 2006).

[30] Vgl. Martin Wilz. Aspekte der Kodierung phonetischer Ähnlichkeiten in deutschen Eigennamen. Magisterarbeit. Inst. F. Phonetik, Abt. Phonetik. Univ. Köln. 2005. <http://www.uni-koeln.de/phil-fak/phonetik/Lehre/MA-Arbeiten/magister_wilz.pdf> (Letzte Einsichtnahme 29. Mai 2006).

Übertragen auf die theoretische Informatik handelt es sich um einen so genannten Automaten, also um einen *Anfangszustand*, an dem der Suchende nachzuschlagen beginnt, eine endliche *Menge an Entscheidungspunkten*, wenn er eine neue Seite aufgeschlagen hat, eine *Übergangsfunktion*, die von jedem Entscheidungspunkt aus die möglichen nächsten Seiten bezeichnet, die in einem Telefonbuch aufgeschlagen werden können, eine Markierung der *Endzustände*, also der Treffer, bei denen ein weiteres Nachschlagen definitiv nicht mehr möglich ist, oder ersichtlich wird, dass der gesuchte Name nicht im Telefonbuch enthalten ist, und eine endliche Menge an Eingabezeichen, dem sogenannten *Alphabet*. Es gilt anzumerken, dass dieser Terminus weniger die Ordnung der Zeichen, als vielmehr die Menge der möglichen Eingabezeichen selbst darstellt.

Letzteres wird besonders wichtig, wenn wir nun die Nachschlagemethode nicht als Entscheidungsbaum anhand einer etwa alphabetischen, universellen und linearen Ordnung aufbauen und stets nur zwischen zwei möglichen weiteren Nachschlageoperationen unterscheiden, sondern wenn es viele so genannte Übergänge gibt. Das heißt, wenn also unser Entscheidungsmechanismus an jedem Zustand einen neuen Buchstaben erhält. Dies ist vergleichbar mit einem Vorgehen in einem Telefonbuch, bei dem wir nicht in der Mitte anfangen, sondern direkt einen bestimmten Buchstaben anspringen. Nehmen wir an, es sei der Eintrag „Huberhans“ gesucht. Da wir nun wissen, dass alle Namen unter „H“ mit diesem ersten Buchstaben anfangen, können wir uns auf das zweite Zeichen stürzen: „u“. Wenn wir annehmen, dass alle zweiten Buchstaben innerhalb der Menge der Namen mit „H“ ebenfalls indexiert sind, können wir direkt die Sektion mit dem zweiten Buchstaben „u“ anspringen. Die übrigen Indexeinträge, welche zu anderen zweiten Buchstaben führen, können wir getrost vernachlässigen.[31] Ein derartiges Verfahren wird Trie[32] genannt. Da sich in unserem Nachschlagekonstrukt keine Rekursionen befinden, und die Übergänge stets tiefer in das Telefonbuch hineinführen, nennt man es auch einen gerichteten azyklischen Graphen (directed acyclic graph, „DAG“). DAGs sind die grundlegenden Datenstrukturen für jede moderne Indexierungstechnik. Ein Suffix Tree ist eine Variante eines Tries, der die Feststellung erlaubt, ob ein bestimmter String ein Teilstring eines anderen Strings ist.

Die genannten Verfahren zur Feststellung einer Ähnlichkeit zwischen zwei Zeichenketten wurden ursprünglich als lineare Verfahren spezifiziert – und zwar sowohl die mathematischen, als auch die phonetischen Verfahren. Anforderungen aus dem Information Retrieval, bei denen der gesuchte String nicht, dafür aber eine klare Definition von Ähnlichkeit vorgegeben ist, führten zu Verfahren, ähnliche Treffer in einer Schlüsselmenge zu identifizieren. Die

[31] Eine exzellente Einführung in die Automatentheorie stammt von John Hopcroft: John E. Hopcroft et al. Introduction to Automata Theory, Languages and Computation. 2nd ed. Upper Saddle River, N.J.: Pearson, 2003. S. 37 ff.

[32] Von „Retrieval“.

schnelle Erschließung der Schlüsselmenge, die womöglich Millionen an Schlüsseln (Wörter) enthalten kann, ist Voraussetzung für eine Anwendbarkeit der Verfahren im Information Retrieval. Manber und Myers sowie Cobbs haben gezeigt, dass lineare Approximationsverfahren auch mit Indexbasierten Erschließungsansätzen implementierbar sind. [33]

3.2. Approximative Indexierung unter Verwendung eines nichtphonetischen Verfahrens

Viele der hier verwendeten Verfahren stammen aus der OCR (Optical Character Recognition): Es ist ein potentiell fehlerbehaftetes Eingabewort gegenüber einer normativen Wortliste abzugleichen. Aus dem OCR-Kontext wird auch ersichtlich, dass die hier verwendeten Algorithmen jegliche Fehler korrigieren können müssen. Phonetische Verfahren sind in einem derartigen Aufgabenkontext nicht sinnvoll, denn nicht das Eingabewort ist normativ, sondern die ähnlichste Zeichenkette im Lexikon.

Das Verfahren besteht darin, den Korpus so weit vorzuverarbeiten, dass zum Zeitpunkt der Anfrage keine linearen Suchoperationen mehr nötig sind, sondern nur noch in einem Index nachgeschlagen werden muss. Die Liste aller im Korpus vorkommenden Wörter wird in Form eines Automaten gespeichert.[34]

Während der Erstellung des Indexes – und damit deutlich vor dem Zeitpunkt der Abfrageverarbeitung – werden für die benötigten Editierdistanzgrenzen universelle, deterministische Levenshtein-Kontrollautomaten generiert, die dazu dienen, im Durchlauf das Auftreten von Fehlern und das Überschreiten der maximalen Fehlergrenze anzuzeigen.[35]

Wenn eine Anfrage bearbeitet werden muss, beginnt man in beiden Automaten im Startzustand und durchwandert Übergang für Übergang, solange die einzelnen Zeichen der eingegebenen Zeichenkette im Wörterbuch-Automaten gefunden werden. Sobald ein Eingabezeichen nicht mehr im Wörterbuch gefunden werden kann, wird im Kontrollautomaten ein Übergang gewählt, der auf einen Zweig führt, dessen Zustände alle eine Abweichung vom Eingabe-

---

[33] U. Manber & E. W. Myers. Suffix Arrays: A new method for on-line string searches. SIAM Journal on Computing, 22(5), 1993. S. 935-948. Siehe auch: A.L. Cobbs. Fast approximate matching using suffix trees. In: CPM95, Lecture Notes in Computer Science, vol. 937. Springer, Berlin Heidelberg NewYork, 1995. S. 41-54.

[34] Vgl. Daciuk, Jan & Stoyan Mihov & Bruce W. Watson & Richard E. Watson. 2000. Incremental construction of minimal acyclic finite-state automata. Computational Linguistics, 26(1). S. 3-16.

[35] Zur Erstellung von universellen deterministischen Levenshtein-Automaten und deren Abarbeitung vgl.: Stoyan Mihov & Klaus U. Schulz: Fast Approximate Search in Large Dictionaries. Computational Linguistics, Volume 30, Issue 4, Dezember 2004. S. 451-477. <http://lml.bas.bg/~stoyan/J04-4003.pdf> (Letzte Einsichtnahme: 30.05.2006).

string kodieren. Im Wörterbuch-Automat folgt man dem ersten, abweichenden Zeichen.

Falls in beiden Automaten ein Endzustand erreicht wird, befindet sich das im Wörterbuch-Automaten gefundene Wort innerhalb der vorgegebenen Toleranzgrenze der Editierdistanz.

Falls im Kontrollautomat festgestellt werden muss, dass die Editierdistanzgrenze erreicht wurde, wird ein Backtracking-Verfahren eingeleitet und die Suche wird an dem Knoten fortgesetzt, an dem die erste Abweichung toleriert wurde, sofern an diesem Knoten weitere Übergänge eine Fortsetzung zulassen.

3.3. Verwandte Arbeiten

Jan Strunk hat ein Verfahren für das orthographisch nicht normierte Plattdeutsch vorgestellt.[36] Zur Aufweichung des Recall verwendet er ebenfalls eine Levenshtein-Distanzberechnung. Um jedoch die Precision zu erhöhen, verwendet er die Levenshtein-Systematik nicht als Automat, sondern als Transduktor, indem er bestimmten Übergängen eine geringere Kostenfunktion zuweist: So erhalten Übergänge für Zeichen, die häufig ausgelassen werden oder phonetisch schwach ausgeprägt sind, statt einer ganzen Distanzoperation nur den halben Distanzwert. Substitutionen werden nur mit 0.25 Distanzzählern gewertet, falls die beiden Zeichen aus derselben graphemischen Klasse stammen.

Zudem erfolgt die Ablage der Wörter im Index in einer syllabischen Transkription, also ebenfalls in einer Abstraktionsform, die vokalische Grapheme mit einer 1 und konsonantische Grapheme mit einer 2 ersetzt.

Methodisch ist dieses Verfahren hinsichtlich der Transkription und der Verwendung eines mit einer Kostenfunktion versehenen Levenshtein-Distanzmaßes höchst interessant. Die phonetische Approximation erfolgt dabei gekoppelt mit dem mathematischen Distanzmaß in einem gewichtenden Transduktor.

Nachteilig hinsichtlich der Verarbeitungsgeschwindigkeit dürfte sich die Tatsache auswirken, dass die Levenshtein-Distanz linear berechnet wird: Nachdem alle syllabischen Kandidaten aus dem Index gewonnen wurden, werden diese linear auf eine vorgegebene Editierdistanzgrenze geprüft. Zudem bietet die Kombination der Approximationsgewichte mit der Transduktorkompilierung den Nachteil der vergleichsweise hohen Komplexität. Dadurch, dass zwei Approximationsverfahren gemischt werden, können die Effekte jedes einzelnen Verfahrens nur sehr schwer vorhergesagt werden.

---

[36] Vgl. Jan Strunk: Information retrieval for languages that lack a fixed orthography, 2003 <http://www.linguistics.ruhr-uni-bochum.de/~strunk/LSreport.pdf> (Letzte Einsichtnahme: 21.4.2006). Von Jan Strunk stammt ein Indexer für plattdeutsche Texte, der auf der Lucene-Engine aufbaut und damit unstrukturierte Texte indexiert. Da „DING“ das Ziel hat, einen Indexer für semistrukturierte Texte zu entwickeln, erweist sich Lucene leider als ungeeignet. Vgl. <http://lucene.apache.org/> (Letzte Einsichtnahme: 30.05.2006).

Angesichts der geringen Zeitvorgabe für die vorgestellte Arbeit ist die Wahl eines linearen Berechnungsverfahrens für die Editierdistanz verständlich. Ein strukturelles Prüfverfahren wäre nach aktuellem Stand vermutlich der Weg der Wahl. Allerdings würde ein derartiger Architekturansatz den hier gewählten Einsatz von Lucene überflüssig erscheinen lassen und zu einem deutlichen Mehraufwand in der Softwareentwicklung führen.

4. Approximative Indexierung unter Verwendung phonetischer Verfahren

4.1. Methodische Annahmen

Wir stellen fest, dass Texte aus der gesamten neuhochdeutschen Epoche für moderne Leser im Allgemeinen gut verständlich sind. Das bedeutet, dass sich die Aussprache größerer Sprachräume, abgesehen von kleinen Dialekträumen, nicht wesentlich verändert hat.[37]

Wir stellen ebenfalls fest, dass die Schreibung sich im Wesentlichen bis zur Berliner Orthographischen Konferenz 1901 verändert hat, aber die Lautung im Zusammenhang mit dem Sprachgebrauch der Kanzleien der frühen Neuzeit gewissen Normierungsentwicklungen unterworfen war und sich in Bezug auf die entstehende Hochsprache nicht mehr wesentlich verändert hat.[38] Nach einem Überblick von Ulrich Kriegesmann über den Forschungsstand der diachronen germanistischen Linguistik ist festzustellen, dass im Allgemeinen von der homogenisierenden Wirkung der Schriftsprache ausgegangen werden kann, die der Homogenisierung der gesprochenen Sprache vorausgeht.[39]

Unter der Annahme, dass ein Verfahren zur Erkennung von ähnlichen Namensvarianten auf dem Hintergrund der genealogischen Forschung geeignet ist, alte Schreibungen zu identifizieren, indem auf eine phonetische Repräsentation geschlossen wird, und die Repräsentation der alten und der neuen Schreibung zumeist übereinstimmen, gehen wir davon aus, dass ein derartiges phonetisches Abstraktionsverfahren in der Lage ist, alte und neue Schreibungen einander zuzuordnen.

---

[37] Gerhard Wolff. Deutsche Sprachgeschichte. 3. Aufl. UTB 1581, Francke, 1994. S. 103: „Damit ist eine weitgehende Verstehbarkeit frühneuhochdeutscher Texte für den heutigen Leser gegeben, so dass wir auf Übersetzungen verzichten können."

[38] Vgl. Stefan Sonderegger. Grundzüge deutscher Sprachgeschichte. Diachronie des Sprachsystems. Bd. 1. Einführung – Genealogie – Konstanten. Berlin/New York, 1979. Kap. 5.2: „Konstante Entwicklungstendenzen in der Geschichte der deutschen Sprache". S. 219-319.

[39] Ulrich Kriegesmann. Die Entstehung der neuhochdeutschen Schriftsprache im Widerstreit der Theorien. Frankfurt a.M., Bern, New York, Paris, Peter Lang, 1990. S. 277 ff.

Da wir dennoch keine stets übereinstimmenden Ergebnisse, sondern auch hier verschiedene Differenzen erwarten,[40] kombinieren wir die bisherigen Abstraktionsansätze mit dem oben beschrieben Verfahren, in einem Index an tatsächlich im Index vorkommenden Zeichenketten die „nearest neighbours“ zu identifizieren.

4.2. Approximation und Indexierung in Datenbanken

Grundsätzlich besteht die Möglichkeit, zum Zeitpunkt der Befüllung eines Datenkonstrukts, also eines Index oder einer Datenbank, die einzelnen Zeichenketten einer phonetischen Reduktion zu unterwerfen. Die reduzierten phonetischen Codes der Wörter werden in einer Tabelle gespeichert, und es ist möglich, über einem derartigen Datenbankfeld einen Index aufzubauen.

Tatsächlich wurde bislang häufig auf diese Weise verfahren, wenn eine größere Textmenge approximativ indexiert werden musste. Das oben erwähnte Verfahren „Phonet“ wurde ursprünglich als eine dBase-Funktion vorgestellt und liegt in der Version 1.3.3 (18. Januar 2002) auch als „stored procedure“ für die Datenbank Postgres vor.[41] Allerdings weisen die Indexierungsmechanismen der gängigen Datenbanken keine Approximationsfunktionen auf, weshalb mit diesem Vorgehen lediglich eine reine phonetische Suche ohne weitere Fehlertoleranz implementierbar ist.

Die Verwendung von Datenbanken zur Indexierung unterschiedlich tief ausgezeichneter, heterogener XML-Korpora stellt jedoch ein grundsätzliches Problem dar[42], und so ist es angebracht, Indexierungsverfahren ohne Verwendung einer Datenbank zu untersuchen, wobei diese um leistungsfähige Konstrukte zur Abbildung der historischen deutschen Phonotaktik erweiterbar sein müssen.

---

[40] Vgl. Sonderegger. Grundzüge. Kap. 5.3: „Inkonstante Merkmale in der Geschichte der deutschen Sprache“. S. 319-353. Vgl. auch: Frédéric Hartweg &. Klaus-Peter Wegera: Frühneuhochdeutsch. Eine Einführung in die deutsche Sprache des Spätmittelalters und der frühen Neuzeit, 2. überarb. Aufl., Tübingen 2005 (Germanistische Arbeitshefte 33). S. 133-149.

[41] Unter der folgenden URL wird die phonet-Version als Stored Procedure für Postgres zum Download angeboten: <http://www.icsy.de/software/phonet/pg_phonet.tar.gz> (Letzte Einsichtnahme: 30. Mai 2006).

[42] Durch die Entwicklung von nativen XML-Datenbanken ergeben sich jedoch Lösungsansätze – allerdings bislang ohne approximative Funktionen. Vgl.: H. V. Jagadish & Shurug Al-Khalifa & Laks Lakshmanan & Andrew Nierman & Stylianos Paparizos & Jignesh Patel & Divesh Srivastava & Yuqing Wu. Timber: A native XML database. Technical report, University of Michigan, April 2002. <http: //www.eecs.umich.edu/db/timber/> (Letzte Einsichtnahme: 30.05.2006).

4.3. Phonet2 – Ein Phonetischer Transduktor von Jörg Michael

Phonet2 ist ein Algorithmus, der ein Eingabewort über kontextsensitive Transformationsregeln in zwei mögliche phonetische Repräsentationen übersetzt. Die erste Repräsentation ist etwas strikter und die zweite Repräsentation zeichnet sich durch eine etwas freiere Varianz aus. In der aktuellen Version 1.3.4[43] vom 14. August 2003 liegen dem Programm mehr als 950 Regeln bei. Im Unterschied zur Version 1.3.1 sind diese nun in der Header-Datei ausgelagert und können recht angenehm editiert werden. Phonet bietet eine Konsistenzprüfung der Regeln auf Konflikte und Widersprüche, jedoch sollte man bei Änderungen stets Vorsicht walten lassen, da viele der Regeln aufeinander aufbauen. Der Autor hat für die Gestaltung des Regelwerks eine eigene Regelsprache entwickelt, mit der Gruppierungen, Wortanfänge und -enden und verschiedene Eigenschaften von Eingabewörtern adressiert werden können.

Die zahlreichen Regeln werden seit der Version 1.3 mit einem Hash angesprungen, wodurch die Zugriffslatenz im Vergleich zur Vorversion auf ein Drittel reduziert werden konnte. Unsere Messungen am Gesamtprogramm ergaben eine durchschnittliche Gesamtlaufzeit von 2 Millisekunden, wobei das Laden der Regeln sowie die Ausgabe des Ergebnisses auf dem Bildschirm eingeschlossen waren. Wir gehen davon aus, dass der Einsatz von phonet als Bibliothek ohne ein erneutes Laden bei jedem Aufruf sowie in der reduzierten Form ohne Ein- und Ausgabe zu weiteren deutlichen Geschwindigkeitsgewinnen führen kann. Unsere Messungen erfolgten unter Linux auf einem AthlonXP-Prozessor mit 1400 MHz. Die verwendete Version (1.3.4) ließ sich mit dem GNU C-Compiler 4.0.4 problemlos und ohne Warnungen kompilieren.

4.4. Konzeptionelle Integration

Eine reine Levenshtein-Approximation auf den primären Textstücken hat zwei Probleme zu bewältigen: Einerseits muss bei Texten aus dem Zeitraum vor der Verfügbarkeit orthographischer Regeln mit einer breiten Varianz an Schreibungen gerechnet werden. Bereits diese Varianz muss durch Zähler im Distanzmaß abgefangen werden. Bedenkt man, dass das Levenshtein-Distanzmaß unter approximativer Verwendung üblicherweise nur bis zu einer Distanz von drei Operationen sinnvolle Ergebnisse liefert, so wird klar, dass man mit Distanzzählern „sparsam“ umgehen muss. Andererseits besteht die Aufgabe eines approximativen Ansatzes darin, verschiedene Aussprachevarianten zu konsolidieren.

Wir schlagen nun folgenden Ansatz vor: Bei dem oben beschriebenen Verfahren von Mihov und Schulz muss der zu indexierende Korpus zuerst tokeni-

[43] <ftp://ftp.heise.de/pub/ct/listings/phonet.zip> (Letzte Einsichtnahme: 30. Mai 2006).

siert werden. Da dies zu einem nicht-zeitkritischen Zeitpunkt erfolgt, ist die ohnehin gute Performanz des Phonet-Verfahrens hier nicht entscheidend. Eine Translation der Zeichenketten aus dem Korpus würde also zu einem Index an phonetischen Repräsentationen führen. Zur Anfragezeit müsste nun das Eingabewort einmal ebenfalls umgerechnet werden und würde dann als phonetisch bereinigte Vorlage zur Suche in einem ebenfalls phonetisch bereinigten Index dienen.

4.5. Experimentelle Überprüfung

In einem Experiment wurden 790 Wörter der Anfangsbuchstaben A-C aus der Wortliste des frühneuhochdeutschen Texts „Till Eulenspiegel“[44] mit ihren modernsprachlichen Schreibungen versehen. Daraufhin wurden die Wortpaare mittels „phonet2“ in phonetische Repräsentationen überführt. Diese wiederum wurden mit einer Levenshtein-Distanzfunktion verglichen. An den Phonet-Regeln wurden keine Modifikationen hinsichtlich der diachronen Veränderungen der Sprache vorgenommen.

Die Statistik zeigt, dass 688 Wortpaare innerhalb einer Distanzmaßgrenze von drei Operationen (0,1,2,3) und 122 Wortpaare außerhalb dieser Grenze liegen. Damit ergibt sich unter der Annahme, dass ein nichtlinearer, Trie-basierter Suchansatz mit einem Kontrollautomaten ebenfalls in der Lage ist, die innerhalb der Distanzmaßgrenze liegenden Varianten zu finden, ein Recall von 0.87. Bei einer Distanzmaßgrenze von kleiner als zwei Operationen liegen 380 Wortpaare innerhalb der Maximaldistanz und 410 außerhalb – Ein Recall-Wert von 0.481.

Die Precision beschreibt hier das Verhältnis der Anzahl der als relevant bewerteten Treffer zur Anzahl der insgesamt gefundenen Treffer. Wir werten das Verfahren wie folgt aus: Für alle modernsprachlichen und für alle historischen Schreibungen wird die entsprechende phonetische Abstraktion errechnet. Sodann wird jeder der gewonnenen modernsprachlichen Abstraktionscodes mit jedem der historischen Codes verglichen. Falls eine Editierdistanz kleiner zwei oder kleiner als die Länge des Code minus zwei auftritt, so wird die historische Schreibweise sowie die moderne Variante als ein Wortpaar vermerkt. Die Berücksichtigung der Länge des phonetischen Codes spielt eine sehr große Rolle, da bei Codes mit zwei Zeichen eine Distanz von zwei beliebigen anderen Codes als passend markieren würde.

Es entsteht also ein „Hash of Lists“, wobei der Schlüssel das Paar mit der jeweils alten und neuen Schreibung ist, und die Liste aus ebensolchen Paaren

[44] Alexander Weissenhorn. Eyn wunderbarliche vnd seltzame History/ von Dyll Vlnspiegel/ bürtig auß dem land Brunschweig/ wie er sein leben verbracht hatt/ newlich auß Saechsicher sprach auff gut Teutsch verdolmetschet/ seer kurtzweilig zulesen/ mit schoenen figuren. Augsburg, 1540.
Vgl. <http://eulenspiegel.is.guad.de/> (Letzte Einsichtnahme: 30.05.2006).

besteht. Anschließend wird manuell untersucht, wie viele der anhand der phonetischen Distanz gefundenen Worte völlig unzutreffend und anhand des Wortsinnes zutreffend sind. Die Summe der relevanten gefundenen zu der Summe aller gefundenen Wörter ergibt den Precision-Wert. Wir kommen bei der oben definierten Editierdistanzgrenze auf eine Precision von 0,598 (590 von 986 Zuordnungen als treffend).

Verglichen mit den Ergebniswerten des Verfahrens von Ernst-Gerlach und Fuhr zeigt sich, dass unsere Ansätze ohne jegliche Optimierungen durchaus auf vorzeigbare Ergebnisse kommen. Allerdings haben Ernst-Gerlach und Fuhr sich auf Texte des 19. Jahrhunderts konzentriert. Unser Versuch erfolgte mit einem Text des Jahres 1540. Wir haben beobachtet, dass eine Optimierung der Transkriptionsregeln, zusammen mit der Bedingung der Identität im Präfix zu einer deutlichen Steigerung des Ergebnisses führen müsste.

5. Ausblick und weitere Arbeiten

Es muss angemerkt werden, dass die Versuchsanordnung der vorgenommenen Experimente nicht von neutralen Beteiligten, sondern in der Natur des Experiments in einem Nachweisinteresse hinsichtlich der Eignung der Methode erfolgten. Aus diesem Grund betrachten wir die gewonnenen Erkenntnisse kritisch.

Allerdings kann anhand der durchweg guten Ergebnisse und aufgrund der Schlüssigkeit des Ansatzes sehr wohl von einem Indiz gesprochen werden, dass die Implementierung eines Schulz-Mihovschen approximativen Indexers auf Basis einer phonetischen Abstraktion für Texte der neuhochdeutschen Sprachepoche insgesamt gute Ergebnisse erwarten lässt.

Bislang konzentrierten sich die Entwicklungsarbeiten an unserem Projekt „DING“ auf die Dokumentenvorverarbeitung und die Befüllung des Index, wobei wiederum der Schwerpunkt auf den XML-spezifischen Verarbeitungsschritten lag. Zum Zeitpunkt der Kalenderwoche 21 / 2006 sind der Feeder samt XML-Parser, die Architektur der DocumentProcessing Pipeline, der DataGuide und erste Teile des WordIndex fertiggestellt. Die nächsten Schritte bestehen in der Implementierung der Lookup-Verfahren nach der Spezifikation von Mihov und Schulz.

Literatur

<ftp://ftp.heise.de/pub/ct/listings/phonet.zip> (Letzte Einsichtnahme: 30.05. 2006). Aktuelle Downloadquelle für „Phonet“.

<http://aspell.net/metaphone/> (Letzte Einsichtnahme 26.05.2006). Eine Überblicksseite über Lawrence Philips MetaphoneAlgorithmus.

<http://lucene.apache.org/> (Letzte Einsichtnahme: 30.05.2006). Die Projektseite der Lucene-Indexierungsbibliothek.

<http://patft.uspto.gov/netacgi/nph-Parser?patentnumber=1,261,167> (Letzte Einsichtnahme: 30.05.2006). Patentschrift zu Soundex.

<http://patft.uspto.gov/netacgi/nph-Parser?patentnumber=1,435,663> (Letzte Einsichtnahme: 30.05.2006). Weitere Patentschrift zu Soundex.

<http://www.glue.umd.edu/~oard/research.html> (Letzte Einsichtnahme: 23.05.2006). Doug Oards Linksammlung für Themen rund um Cross-Language Information Retrieval.

<http://www.icsy.de/software/phonet/pg_phonet.tar.gz> (Letzte Einsichtnahme: 30.05.2006). Die Transduktorfunktion „Phonet" als Stored Procedure für Postgres.

<http://wwwuser.gwdg.de/~mthalle2/> (Letzte Einsichtnahme: 30.05.2006): Kleio on the Web.

A. Apostolico & C. Guerra. The Longest Common Subsequence problem revisited. Algorithmica 2, 1970. S. 315-336.

Andrew Binstock & John Rex. Metaphone: A Modern Soundex. Practical Algorithms For Programmers. Addison-Wesley, 1995. S. 160-169.

D. De Bron & M. Olsen. The Guth Algorithm and the Nominal Record Linkage of Multi-Ethnic Populations. Historical Methods, 19 (1986). S. 20-24.

S. Burkhardt & A. Crauser & P. Ferragina & H.-P. Lenhof & E. Rivals & M. Vingron. q-gram based database searching using a suffix array. In: RECOMB99, ACM, NewYork, 1999. S. 77-83.

A.L. Cobbs. Fast approximate matching using suffix trees. In: CPM95, Lecture Notes in Computer Science, vol. 937. Springer, Berlin, Heidelberg, New York, 1995. S. 41-54.

Daciuk, Jan & Stoyan Mihov & Bruce W. Watson & Richard E. Watson. 2000. Incremental construction of minimal acyclic finite-state automata. Computational Linguistics, 26(1). S. 3-16.

Fred J. Damerau. A Technique for Computer Detection and Correction of Spelling Errors. In. Communications of the ACM. Vol 7, Nr. 3 (03/1964). S. 171-176. <http://dalcin.org/referencias/pdf/325.pdf> (Letzte Einsichtnahme 30.05.2006).

G. Das & R. Fleisher & L. Gasieniek & D. Gunopulos & J. Karkainen. Episode matching. In Proceedings of the 8th Annual Symposium on Combinatorial Pattern Matching (CPM '97). LNCS, vol. 1264, Springer-Verlag, Berlin, 1997. S. 12-27.

Andrea Ernst-Gerlach & Norbert Fuhr. Generating Search Term Variants for Text Collections with Historic Spellings. 28th European Conference on Information Retrieval Research (ECIR 2006). <http://www.is.informatik.uni-duisburg.de/bib/pdf/ir/Ernst_Fuhr:06.pdf> (Letzte Einsichtnahme: 30.05.2006).

Gloria J.A. Guth. Surname Spellings and Computerized Record Linkage. Historical Methods Newsletter, 10 (1976). S. 10-19.

Richard Hamming: R.W. Hamming. Error detecting and error correcting codes. Bell System Tech. J. 29, 1950. S. 147-160.

Frédéric Hartweg & Klaus-Peter Wegera: Frühneuhochdeutsch. Eine Einführung in die deutsche Sprache des Spätmittelalters und der frühen Neuzeit, 2. überarb. Aufl., Tübingen 2005 (Germanistische Arbeitshefte 33).

John Hopcroft: John E. Hopcroft et al. Introduction to Automata Theory, Languages and Computation. 2nd ed. Upper Saddle River, N.J.: Pearson, 2003. S. 37 ff.

H. V. Jagadish & Shurug Al-Khalifa & Laks Lakshmanan & Andrew Nierman & Stylianos Paparizos & Jignesh Patel & Divesh Srivastava & Yuqing Wu. Timber: A native XML database. Technical report, University of Michigan, April 2002. <http: //www.eecs.umich.edu/db/timber/> (Letzte Einsichtnahme: 30.05.2006).

Donald E. Knuth, The Art of Computer Programming. Vol. 3, 2nd ed. Boston: Addison-Wesley, 1998. S. 394-395.

Ulrich Kriegesmann. Die Entstehung der neuhochdeutschen Schriftsprache im Widerstreit der Theorien. Frankfurt a.M., Bern, New York, Paris, Peter Lang, 1990.

A.J. Lait & Brian Randell. An Assessment of Name Matching Algorithms. <http://www.hpdd.de/download/wb/20010416.NameMatching.pdf> (Letzte Einsichtnahme: 29. Mai 2006).

Vladimir I. Levenshtein: Binary codes capable of correcting deletions, insertions, and reversals, in: Doklady Akademii Nauk SSSR 163 (1965), S. 845-848 (Russisch). Englische Übersetzung in: Soviet Physics Doklady, 10 (1966), S. 707-710.

U. Manber & E. W. Myers. Suffix Arrays: A new method for on-line string searches. SIAM Journal on Computing, 22(5), 1993. S. 935-948.

Jörg Michael. Doppelgänger gesucht. Ein Programm für die kontextsensitive phonetische Stringumwandlung. In: c't Magazin für Computer und Technik 25 (1999). S. 252 ff. <http://www.heise.de/ct/ftp/99/25/252/> (Letzte Einsichtnahme: 29.05.2006).

Jörg Michael. Nicht wörtlich genommen – Schreibweisentolerante Suchroutinen in dBase implementiert. In: c't Magazin für Computer und Technik 10 (1988). S. 126-131.

Stoyan Mihov & Klaus U. Schulz: Fast Approximate Search in Large Dictionaries. Computational Linguistics, Volume 30, Issue 4, Dezember 2004. S. 451-477. <http://lml.bas.bg/~stoyan/J04-4003.pdf> (Letzte Einsichtnahme: 30.05.2006).

Gary Mokotoff & Sallyann Amdur Sack. Where Once We Walked. Avotaynu, 2002. S. 567-569. Im Internet: <http://www.avotaynu.com/soundex.html> (Letzte Einsichtnahme 27.05.2006).

Gonzalo Navarro. A Guided Tour to Approximate String Matching. ACM Comput. Surv. 33, 1 (03/2001). S. 31-88.

Gonzalo Navarro & Mathieu Raffinot. Flexible Pattern Matching in Strings. Practical On-Line Search Algorithms for Texts and Biological Sequences. Cambridge: CUP, 2002.

S. Needleman & C. Wunsch. A general method applicable to the search for similarities in the amino acid sequences of two proteins. Journal of Molecular Biology 48, 1970. S. 444-453.

Adam Nelson. Implement Phonetic („Sounds-like“) Name Searches with Double Metaphone. <http://www.codeproject.com/string/dmetaphone1.asp> (Letzte Einsichtnahme 26.05.2006).

Lawrence Philips. Hanging on the Metaphone. Computer Language, Vol. 7, No. 12 (December), 1990.

Lawrence Philips. The Double Metaphone Search Algorithm. C/C++ Users Journal, June 2000. <http://www.ddj.com/dept/cpp/184401251> (Letzte Einsichtnahme 26.05.2006).

H-J. Postel, Die Kölner Phonetik – Ein Verfahren zur Identifizierung von Personennamen auf der Grundlage der Gestaltanalyse in IBM-Nachrichten 19, 1969. S. 925-931.

Rainer Schnell & Tobias Bachteler, Stefan Bender. A Toolbox for Record Linkage. Austrian Journal of Statistics 33 (2004), Nr. 1 & 2. S. 125-133.

Stefan Sonderegger. Grundzüge deutscher Sprachgeschichte. Diachronie des Sprachsystems. Bd. 1. Einführung – Genealogie – Konstanten. Berlin/New York, 1979.

Jan Strunk: Information retrieval for languages that lack a fixed orthography, 2003 <http://www.linguistics.ruhr-uni-bochum.de/~strunk/LSreport.pdf> (Letzte Einsichtnahme: 21.4.2006).

Taft, R.L. Name Searching Techniques. Albany, N.Y.: Bureau of System Development, 1970.

Esko Ukkonen. Approximate string matching with q-grams and maximal matches. Theoretical Computer Science, Vol. 92, Iss. 1, 1992. S. 191-211.

Hans-Peter von Reth & Hans-Jörg Schek. Eine Zugriffsmethode für die phonetische Ähnlichkeitssuche. Technical Report Nr. 77.03.002. Heidelberg: IBM Scientific Center, 1977.

Martin Wilz. Aspekte der Kodierung phonetischer Ähnlichkeiten in deutschen Eigennamen. Magisterarbeit. Inst. F. Phonetik, Abt. Phonetik. Universität zu Köln. 2005. <http://www.uni-koeln.de/phil-fak/phonetik/Lehre/MA-Arbeiten/magister_wilz.pdf> (Letzte Einsichtnahme 29. Mai 2006).

Gerhard Wolff. Deutsche Sprachgeschichte. 3. Aufl. UTB 1581, Francke, 1994.

Justin Zobel & Philip Dart. Finding Approximate Matches in Large Lexicons. Software-Practice and Experience, Vol. 25(3), 1995. S. 331-345.